

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.
No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.
See full “License and Conditions of Use” notice for details.
In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.
Author: Carlos Daniel Borrego Romero

Annex C. General Reference Architecture for Institutional Integration of the LensBalance Model and Its Explainability Layer

Purpose

This annex proposes a general reference architecture in Python to facilitate institutional integration of the LensBalance Model within current ophthalmology workflows, with explicit support for interoperability, multimodal data fusion, clinical decision support, and model explainability.[file:128][web:187][web:196] The architecture is designed to be realistic with current technology in 2026 and to remain modular so that institutions can adapt it to local imaging devices, EHR environments, laboratory systems, and AI infrastructure.[web:171][web:177][web:199]

C.1 Architectural Objectives

The architecture is designed around five objectives.[file:128][web:177][web:196]

1. **Institutional interoperability:** connect imaging, laboratory, and clinical systems without requiring replacement of existing infrastructure.[web:187][web:191][web:200]
2. **Multimodal computation:** fuse heterogeneous inputs into the LensBalance latent components and derived indices.[file:128][web:177]
3. **Clinical usability:** surface outputs inside workflow tools already used by clinicians, including SMART on FHIR and CDS Hooks-compatible environments.[web:187][web:189][web:192]
4. **Explainability:** provide both global and patient-level interpretation of model outputs using transparent component decomposition and explainable AI methods.[file:128][web:195]
5. **Incremental deployment:** allow a minimal implementation using available imaging and laboratory data, with later expansion to richer biomarker and molecular layers.[file:128]

C.2 General Reference Architecture

A practical institutional deployment can be represented in six layers.[web:171][web:177][web:187]

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.
No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.
See full “License and Conditions of Use” notice for details.
In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.
Author: Carlos Daniel Borrego Romero

C.2.1 Layer 1. Data Acquisition

This layer receives source information from Scheimpflug imaging, AS-OCT, slit-lamp or cataract grading systems, laboratory observations such as HbA1c and oxidative stress panels, and structured clinical metadata from the EHR.[file:128] The acquisition layer may also include molecular data from aqueous humor proteomics or metabolomics when available in advanced research or surgical settings.[file:128][web:133][web:145]

C.2.2 Layer 2. Interoperability and Mapping

This layer converts local data into a common exchange model using FHIR-compatible resources or equivalent structured clinical objects.[web:187][web:196][web:199] SMART on FHIR enables substitutable applications across EHR systems, while FHIR-based clinical reasoning and CDS delivery frameworks allow the same computable logic to be reused across institutions.[web:187][web:189][web:194]

C.2.3 Layer 3. Harmonization and Feature Engineering

This layer performs unit harmonization, device normalization, directional alignment, missing-data handling, and transformation of raw indicators into normalized component variables on $[0,1]$. [file:128] It then computes the latent components O_{ox} , A_{prot} , G_{gly} , D_{struct} , A_{antiox} , R_{proteo} , and H_{micro} , followed by H_{lens} , P_{lens} , B_{lens} , S_{lens} , $P(\text{opacity})$, and ΔB_{lens} . [file:128]

C.2.4 Layer 4. Prediction and Explainability

This layer hosts the LensBalance computation engine and its explainability services.[file:128][web:195] It can incorporate SHAP-based global and local explanations, feature-importance summaries, and deterministic component decomposition so that the model output is interpretable both mathematically and clinically.[web:195]

C.2.5 Layer 5. Clinical Decision Support and Presentation

This layer presents results to users through SMART on FHIR apps, embedded dashboards, or CDS Hooks cards triggered during chart review or imaging interpretation.[web:189][web:192] CDS Hooks is particularly useful because it can launch relevant SMART on FHIR applications at the point of care and deliver contextual cards with recommendations, explanations, or links to detailed views.[web:192]

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.
No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.
See full “License and Conditions of Use” notice for details.
In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.
Author: Carlos Daniel Borrego Romero

C.2.6 Layer 6. Audit, Logging, and Governance

This layer records model version, input completeness, data provenance, explanation snapshots, and clinician interactions for traceability and institutional oversight.[web:196][web:199] It supports retrospective auditing, quality improvement, and safe lifecycle management of the integrated model.[web:196]

C.3 Proposed Python System Design

The Python reference design below is intended as a vendor-agnostic institutional backbone rather than a production prescription.[web:199][web:200] The implementation assumes modern Python services, a REST or event-driven interface, and compatibility with FHIR-oriented middleware.[web:187][web:199]

C.3.1 Core Modules

A practical codebase can be organized into the following modules:[file:128][web:187][web:199]

- `connectors/`: retrieves imaging metadata, laboratory results, and EHR observations from institutional systems.[web:187][web:200]
- `mapping/`: maps local data fields to standardized internal objects and FHIR-compatible structures.[web:191][web:199]
- `normalization/`: applies harmonization and normalization rules to create aligned 0–1 indicator values.[file:128]
- `components/`: computes the seven LensBalance latent components from normalized indicators.[file:128]
- `engine/`: computes H_{lens} , P_{lens} , B_{lens} , S_{lens} , probability, and temporal progression.[file:128]
- `explainability/`: produces deterministic decomposition and SHAP-based local/global explanations.[web:195]
- `cds/`: packages outputs for SMART on FHIR applications and CDS Hooks cards.[web:189][web:192]
- `governance/`: logs provenance, versioning, completeness, and audit metadata.[web:196]

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.
 No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.
 See full “License and Conditions of Use” notice for details.
 In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.
 Author: Carlos Daniel Borrego Romero

C.3.2 Python Reference Example

```

from dataclasses import dataclass
from typing import Dict, List, Optional
import math

@dataclass
class NormalizedInputs:
    O_ox: float
    A_prot: float
    G_gly: float
    D_struct: float
    A_antiox: float
    R_proteo: float
    H_micro: float

@dataclass
class LensBalanceConfig:
    damage_weights: Dict[str, float]
    protection_weights: Dict[str, float]
    epsilon: float = 1e-6
    k: float = 8.0
    theta: float = 1.0

@dataclass
class LensBalanceOutput:
    H_lens: float
    P_lens: float
    B_lens: float
    S_lens: float
    opacity_probability: float
    component_contributions: Dict[str, float]

class LensBalanceEngine:
    def __init__(self, config: LensBalanceConfig):
        self.config = config

    def compute(self, x: NormalizedInputs) -> LensBalanceOutput:
        H_lens = (
            self.config.damage_weights["O_ox"]
            * x.O_ox
            +

```

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.

No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.

See full “License and Conditions of Use” notice for details.

In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.

Author: Carlos Daniel Borrego Romero

```
self.config.damage_weights["A_prot"] * x.A_prot +
self.config.damage_weights["G_gly"] * x.G_gly +
self.config.damage_weights["D_struct"] * x.D_struct
)
P_lens = (
self.config.protection_weights["A_antiox"] * x.A_antiox +
self.config.protection_weights["R_proteo"] * x.R_proteo +
self.config.protection_weights["H_micro"] * x.H_micro
)
B_lens = H_lens / (P_lens + self.config.epsilon)
S_lens = H_lens / (H_lens + P_lens + self.config.epsilon)
opacity_probability = 1 / (1 + math.exp(-self.config.k * (B_lens - self.config.theta)))

component_contributions = {
"O_ox": self.config.damage_weights["O_ox"] * x.O_ox,
"A_prot": self.config.damage_weights["A_prot"] * x.A_prot,
"G_gly": self.config.damage_weights["G_gly"] * x.G_gly,
"D_struct": self.config.damage_weights["D_struct"] * x.D_struct,
"A_antiox": self.config.protection_weights["A_antiox"] * x.A_antiox,
"R_proteo": self.config.protection_weights["R_proteo"] * x.R_proteo,
"H_micro": self.config.protection_weights["H_micro"] * x.H_micro,
}

return LensBalanceOutput(
H_lens=H_lens,
P_lens=P_lens,
B_lens=B_lens,
S_lens=S_lens,
opacity_probability=opacity_probability,
component_contributions=component_contributions,
)
```

This example captures the minimal computational core and illustrates how a normalized multimodal input layer can be converted into clinically interpretable LensBalance outputs.[file:128] In institutional settings, this engine would normally be called by a service layer that receives mapped observations from FHIR connectors or internal databases, stores provenance, and returns results to the EHR-facing application.[web:187][web:189][web:200]

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.
 No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.
 See full “License and Conditions of Use” notice for details.
 In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.
 Author: Carlos Daniel Borrego Romero

C.3.3 Example Service Workflow in Python

```
class LensBalanceService:
    def __init__(self, mapper, normalizer, engine, explainer, repository):
        self.mapper = mapper
        self.normalizer = normalizer
        self.engine = engine
        self.explainer = explainer
        self.repository = repository

    def run_for_patient(self, patient_id: str, observation_bundle: dict) -> dict:
        mapped = self.mapper.transform(observation_bundle)
        normalized = self.normalizer.apply(mapped)
        output = self.engine.compute(normalized)
        explanation = self.explainer.explain(normalized, output)
        self.repository.save(patient_id, mapped, normalized, output, explanation)
        return {
            "patient_id": patient_id,
            "output": output,
            "explanation": explanation,
        }
```

This service-oriented pattern separates interoperability, harmonization, computation, explainability, and persistence, which is essential for maintainability and regulatory traceability.[web:196][web:199]

C.4 Institutional Integration Pathway

C.4.1 Minimal Institutional Deployment

A minimal institutional deployment can operate with Scheimpflug or OCT-derived features, HbA1c, selected oxidative stress indicators, and routine clinical metadata.[file:128] This is sufficient to compute an initial version of A_{prot} , D_{struct} , G_{gly} , O_{ox} , and approximate protection-side variables using calibrated proxies.[file:128]

C.4.2 Intermediate Deployment

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.
No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.
See full “License and Conditions of Use” notice for details.
In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.
Author: Carlos Daniel Borrego Romero

An intermediate deployment adds richer laboratory and inflammatory profiling, allowing stronger estimation of A_{antiox} , R_{proteo} , and H_{micro} .^[file:128] At this stage, LensBalance can become a more reliable multimodal clinical support layer for monitoring and follow-up decisions.^{[file:128][web:177]}

C.4.3 Advanced Deployment

An advanced deployment includes ocular-fluid molecular profiling, robust longitudinal data capture, and prospective evaluation of workflow impact.^{[file:128][web:133][web:145]} This provides the strongest alignment between the theoretical model and directly observed lens-environment biology.^[file:128]

C.5 Explainability Power of the Model

The explanatory strength of LensBalance arises from the fact that it is not only predictive but also structurally interpretable.^[file:128] Its outputs are decomposable into named biological domains, and each output can be traced to a mathematically explicit relationship between damage accumulation and protective reserve.^[file:128]

C.5.1 Deterministic Explainability

At the first level, the model is inherently explainable because H_{lens} , P_{lens} , B_{lens} , and ΔB_{lens} are transparent constructs derived from explicitly defined normalized components.^[file:128] This means that clinicians can immediately identify whether a patient's risk state is being driven more strongly by oxidative burden, glycation, aggregation, structural disorder, reduced antioxidant reserve, impaired proteostasis, or unstable microenvironment.^[file:128]

C.5.2 SHAP-Based Explainability

At the second level, SHAP or similar explainable AI methods can quantify both local and global feature contributions within the multimodal implementation.^[web:195] SHAP is especially useful because it supports patient-level explanation and dataset-level interpretation within the same framework, and is widely recognized as a robust approach for explainable machine learning in ophthalmic contexts.^[web:195] In LensBalance, SHAP can operate at two levels: on the raw indicators used to estimate each component, and on the components themselves when they are fed into downstream risk or grading models.^{[file:128][web:195]}

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.
No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.
See full “License and Conditions of Use” notice for details.
In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.
Author: Carlos Daniel Borrego Romero

C.5.3 Multi-Level Clinical Explanation

A clinically useful explanatory report should contain at least four levels of interpretation:[file:128]

1. **Indicator level:** which observed biomarkers or image features were abnormal.
2. **Component level:** which latent domains contributed most to damage or reduced protection.
3. **State level:** whether the overall lens system is stable, borderline, or decompensating.
4. **Temporal level:** whether the imbalance is static, improving, or accelerating over time.[file:128]

This layered explanatory design is one of the principal institutional advantages of LensBalance, because it enables clinicians, researchers, and auditors to interpret the same output from different but compatible perspectives.[file:128]

C.6 CDS and SMART on FHIR Delivery Concept

A practical institutional implementation can expose LensBalance through a SMART on FHIR app that reads current patient context, retrieves the relevant FHIR resources, and renders the component scores, balance indices, probability outputs, and explanation panels within the clinician workflow.[web:187][web:194][web:200] CDS Hooks can trigger this application whenever relevant events occur, such as cataract grading completion, ophthalmic imaging review, or opening a patient chart with eligible data.[web:189][web:192]

This approach is especially attractive because SMART on FHIR applications are designed to be substitutable across EHR environments, while CDS Hooks can increase contextual utilization of those apps at the point of care.[web:192][web:194] For institutions, this reduces the need for deeply customized user interfaces and improves portability of deployment logic.[web:192]

C.7 Governance, Safety, and Monitoring

Institutional adoption requires more than model computation.[web:196] The reference architecture should therefore log model version, parameter configuration, input completeness, explanation snapshots, and user interactions; provide fallbacks when data are incomplete; and distinguish between research-only outputs and clinically validated outputs.[web:196][web:199] These controls are necessary for safe institutional deployment, ongoing recalibration, and transparent review of model behavior over time.[web:196]

This model and documentation are provided for non-commercial use only under the CC BY-NC-SA 4.0 license.
No commercial use, profiling, or exploitation of IP logs or trade secrets is permitted.
See full “License and Conditions of Use” notice for details.
In case of conflict or ambiguity, the most restrictive interpretation in favor of non-commercial use, user privacy, and protection of trade secrets shall prevail.
Author: Carlos Daniel Borrego Romero

C.8 Conclusions

A Python-based reference architecture for LensBalance integration is achievable today using modular services for interoperability, harmonization, multimodal computation, explainability, and clinical presentation.[web:187][web:189][web:196][web:199] The most practical institutional pathway is to deploy LensBalance as a vendor-agnostic feature-engineering and decision-support layer that connects to existing imaging systems and EHR infrastructure rather than replacing them.[file:128][web:191][web:200]

The explanatory power of the model is a major institutional asset.[file:128] Because the framework combines deterministic systems decomposition with explainable AI methods such as SHAP, it can provide transparent, multi-level clinical interpretation while still supporting modern multimodal predictive workflows.[web:177][web:195] This makes LensBalance well suited for integration in institutions that need not only predictive performance, but also traceability, clinician trust, and practical interoperability.[file:128][web:187]

LICENSE AND CONDITIONS OF USE.

The content is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license:

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

By using this content and model, you agree to:

Provide appropriate credit, include a link to the above license, and indicate if changes were made.

Not use the material or any derivative works for commercial purposes.

Distribute any adaptations or derivative works only under the same CC BY-NC-SA 4.0 license.

Not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Data logging and trade secrets policy.

The design, implementation, and deployment of this model, as well as any derivatives, do not authorize the collection, storage, or exploitation of IP addresses, unique device identifiers, or other technical traceability data for commercial profiling, targeted advertising, or any other economic exploitation.

Any system implementing this model must limit the logging and retention of IP addresses and other identifiable data strictly to what is necessary to:

(a) maintain technical security and service integrity; and

(b) comply with applicable legal obligations (for example, any minimum log retention required by law).

It is prohibited to use this model or its derivatives to:

capture, store, or exploit trade secrets, proprietary know-how, or confidential information of users or third parties for any purpose other than providing the technical service described in this documentation; or train or improve closed, commercial systems using data that contains confidential information of third parties without their prior, explicit, written consent.

---+EOD+---